Machine Learning (2020) 109:1569–1586
https://doi.org/10.1007/s10994-020-05889-1

Double random forest

Sunwoo Han[1] · **Hyunjoong Kim**[2] · **Yung-Seop Lee**[3]

Received: 7 September 2019 / Revised: 4 May 2020 / Accepted: 4 June 2020 / Published online: 2 July 2020

Abstract
Random forest (RF) is one of the most popular parallel ensemble methods, using decision trees as classifiers. One of the hyper-parameters to choose from for RF fitting is the nodesize, which determines the individual tree size. In this paper, we begin with the observation that for many data sets (34 out of 58), the best RF prediction accuracy is achieved when the trees are grown fully by minimizing the nodesize parameter. This observation leads to the idea that prediction accuracy could be further improved if we find a way to generate even bigger trees than the ones with a minimum nodesize. In other words, the largest tree created with the minimum nodesize parameter may not be sufficiently large for the best performance of RF. To produce bigger trees than those by RF, we propose a new classification ensemble method called double random forest (DRF). The new method uses bootstrap on each node during the tree creation process, instead of just bootstrapping once on the root node as in RF. This method, in turn, provides an ensemble of more diverse trees, allowing for more accurate predictions. Finally, for data where RF does not produce trees of sufficient size, we have successfully demonstrated that DRF provides more accurate predictions than RF.

Keywords Classification · Ensemble · Random forest · Bootstrap · Decision tree

Editor: Byron Wallace.

✉ Hyunjoong Kim
hkim@yonsei.ac.kr

Sunwoo Han
shan@fredhutch.org

Yung-Seop Lee
yung@dongguk.edu

[1] Vaccine and Infectious Disease Division, Fred Hutchinson Cancer Research Center, Seattle, WA 98006, USA

[2] Department of Applied Statistics, Yonsei University, Seoul 03722, South Korea

[3] Department of Statistics, Dongguk University, Seoul 04620, South Korea

1 Introduction

The ensemble method combines multiple different models to achieve better prediction accuracy for classification and regression (Dietterich 2000). Ensemble methods generally perform better than single fitted models (Hansen and Salamon 1990). Because of the good performance, ensemble methods are widely used in machine learning and statistics community (Breiman 1996; Freund and Schapire 1996; Bauer and Kohavi 1999; Amaratunga et al. 2008; Wolf et al. 2010). Classification ensemble methods consist of several classifiers, typically decision trees. The well-known methodologies of classification ensemble are Boosting (Freund and Schapire 1996; Schapire 1990; Freund and Schapire 1997), Bagging (Breiman 1996), Random Forest (Breiman 2001), Gradient Boosting (Mason et al. 1999; Hastie et al. 2009), and XGBoost (Chen and Guestrin 2016).

Random forest (RF) is a widely used method in various fields because it has many advantages over other classification ensemble methods. RF is fast in both training and prediction, resistant to label noise and outliers, has multi-class capabilities, is well suited for parallel processing and delivers superior performance for high-dimensional data (Hastie et al. 2013; Friedman et al. 2001).

Choosing appropriate hyper-parameters is critical for improving RF performance. Typical parameters are number of trees (*ntree*), number of candidate features (*mtry*), sample size (*samplesize*), and tree size (*nodesize*). Various studies have been carried out on the effect of different parameters on RF performance. Probst and Boulesteix (2018), Freeman et al. (2015) and Huang and Paul (2016) examined the sensitivity of the parameters. Banfield et al. (2007), Hernandez-Lobato et al. (2013) and Oshiro et al. (2012) tested the effect of the number of trees. Boulesteix et al. (2012) and Han and Kim (2019) proposed methods for selecting the appropriate number of candidate features. Martínez-Muñoz and Suárez (2010) studied on the estimation of sample size for Bagging. Lin and Jeon (2012) developed the idea of combining RF with the adaptive nearest neighbors to estimate the tree size.

This paper is about developing a new ensemble method that surpass RF. The idea stems from the question whether a tree of sufficiently large size is being generated during the RF process. The classification ensemble method proposed in this paper is called DRF, which will be explained in detail in subsequent sections. Section 2 reviews the original RF and provides motivation for this research. The novel algorithm of the new classification ensemble method is described in Sect. 3. In Sect. 4, we extend the previous motivational experiments to compare the performance of DRF and RF in terms of accuracy and tree size. We compare the prediction accuracy of DRF and other classification ensemble methods in Sect. 5. Section 6 concludes this study.

2 Motivation

2.1 Random forest

Random forest (RF) (Breiman 2001) completes the ensemble by creating trees on the bootstrap data. When constructing the trees, the RF algorithm chooses the optimal split among a randomly selected set of features for every intermediate nodes. It is well known

that RF improves prediction accuracy because it produces more diverse and less correlated trees (Breiman 2001; Banfield et al. 2007). Algorithm 1 describes the RF algorithm.

Algorithm 1: The RF algorithm.

Training Phase :
Given :
- D : training set with n instances, p features, and target variable.
- K : number of classes in target variable.
- B : number of classifiers in RF.

Procedure :
For $b = 1$ to B
1. Generate bootstrapped sample D_b^* from the training set D.
2. Grow a tree using a random feature subset from bootstrapped sample D_b^*.
   For a given node t,
   (i) Randomly select $m \approx \sqrt{p}$ or $m \approx p/3$ feature.
   (ii) Find the best split features and cutpoints using the random feature subset.
   (iii) Send down the data using the best split features and cutpoints.
   Repeat (i)-(iii) until stopping rules are met.
3. Construct trained classifiers C_b.

Test Phase :
Aggregate the B trained classifiers using simple majority vote. For a test instance x, the predicted class label from classifiers C_B is given as :

$$C_B(x) = \text{argmax}_j \sum_{b=1}^{B} I(C_b(x) = j), \text{for } \; j = 1, ..., K$$

2.2 The tree size effect on random forest

Trees created with RF are usually made to full size without pruning (Breiman 2001). This is because usually the lower bias is obtained when each tree grows to its maximum size. Along with the variance reduction effect of the ensemble, the bias reduction is expected to ultimately improve the classification accuracy.

In the *randomForest* package of R program, *nodesize* and *maxnodes* parameters control the tree size of RF, where *nodesize* is the minimum number of cases a terminal node should hold and *maxnodes* is the maximum number of terminal nodes allowed in a tree. Thus, if *nodesize* is small and *maxnodes* is large, the tree will grow large. The default value for *nodesize* is 1 in classification and 5 in regression, and *maxnodes* defaults to NULL which means that there is no limit on the maximum number of nodes. It is empirically known that using these defaults yields good RF performance (Huang and Paul 2016).

As a motivational example, an experiment was conducted to investigate the effect of tree size on RF performance using 'mnist' data, which is often used for classification problems in the field of machine learning. The 'mnist' data consist of images of handwritten digits 0–9, represented by a grid of 28×28 pixels. From the modeling point of view, this data is interpreted as having 10 classes and 784 variables. The training and test sets have 60,000 and 10,000 instances, respectively. Figure 1 is an example of the 'mnist' dataset.

The design of the experiment is as follows. Using the approach of Larochelle et al. (2012), we fit and evaluate the RF model. That is, the training data is divided into six pieces, each consisting of 10,000 instances. Five pieces are collected to construct a learning

Fig. 1 Examples of 'mnist' data

sample for fitting, and the original test set is used for evaluation. This process is repeated six times, in which all combinations are formed.

To investigate the effect of tree size on RF performance, various *nodesize* values were used, namely *nodesize* = $(0.1n, 0.09n, \ldots, 0.01n, 0.005n, 0.001n,$ default), where n is the number of instances in the training set, and the default is 1. The number of trees in RF is set to 200 to allow sufficient model fit. Other parameters in the *randomForest* R package were left at their default values.

The result of the experiment was summarized in Fig. 2. The horizontal axis represents *nodesize* value and the vertical axis represents the accuracy of the test data. The boxplot in the figure is drawn using the test accuracy obtained from six repeated experiments.

First, we noticed that RF performance was heavily influenced by the *nodesize* option. Second, as the value of *nodesize* decreases, that is, as the tree size increases, the test accuracy gradually improves. Thus, in the case of the 'mnist' dataset, the best RF performance was when the tree size is maximized. If RF is able to construct even bigger trees, can the prediction accuracy be further improved? If so, the current RF algorithm has a limitation

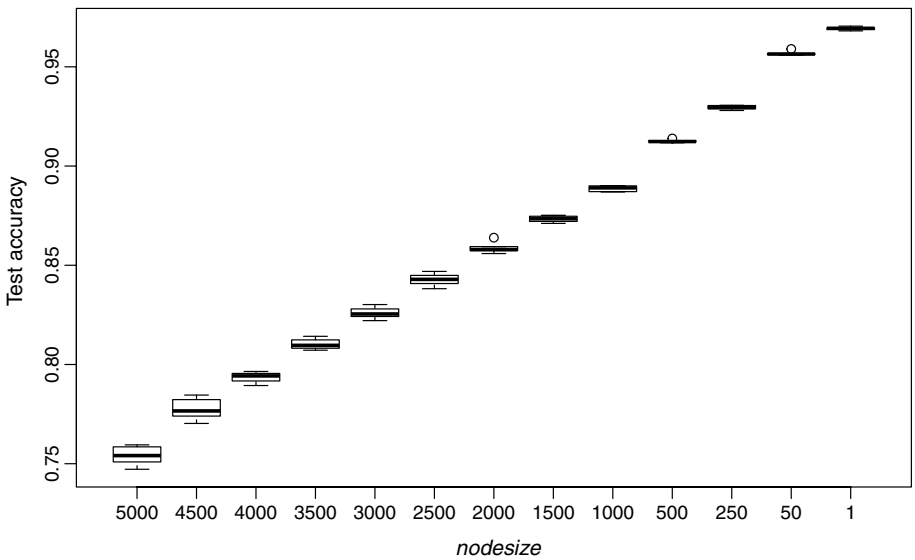

Fig. 2 Box-plot of the test accuracy of random forest for each *nodesize* using 'mnist' data

that the tree size is not large enough for the 'mnist' dataset. In other words, even the largest fitting provided by RF is under-fitting in the 'mnist' dataset. This question motivated us to develop a new classification ensemble method to overcome the limitation of the RF algorithm in terms of tree size.

3 Double random forest (DRF)

We propose a new classification ensemble method called "Double Random Forest" (DRF) to improve the RF performance when RF under-fits the data.

First, DRF uses the whole training data to grow a decision tree at each stage of ensemble. This is in contrast to RF that constructs individual trees using random bootstrap samples from the training set. This means that in DRF, all trees are created with the same data from the beginning, whereas in RF, the trees are created with different data having fewer unique instances. Note that the number of unique instances of the bootstrap sample is about $1 - e^{-1} \approx 0.632$ times of the training data. It is obvious that the more unique instances you have, the easier it is to create larger trees. Therefore, the DRF ensemble tends to consist of larger trees, compared to RF.

Second, DRF uses the bootstrap sampling momentarily for each intermediate node, including the root node, only to find the partitioning rule. In addition, a random subset of the features is also selected as in RF. Finally, DRF searches for the best partitioning rules by using a random bootstrap of instances and a random subset of features on each node of the tree. Once the partitioning rule is found, it recovers the original data in the node and sends the instances down to the child nodes.

In summary, DRF uses more instances to create trees, which increases the size of the trees, thereby reducing the bias in classification prediction. It also adds randomness to the partitioning rules by bootstrap and feature subset, allowing more diverse ensemble of trees.

Using the bootstrap near the terminal node may not be appropriate because there are considerably fewer unique instances. Therefore, we chose not to use bootstrap on nodes with instances less than 10% of the total number of instances. Algorithm 2 explains the DRF algorithm.

Algorithm 2: The DRF algorithm.

Training Phase :
Given :
- D : training set with n instances, p features, and target variable.
- D_t : training set with n_t instances, p features, and target variable at a node t.
- K : number of classes in target variable.
- B : number of classifiers in DRF.

Procedure :
For b = 1 to B
1. Use the training set D.
2. Grow a tree using a random feature subset and a random bootstrap of instances from training set D.
   For a given node t,
   (i) If $n_t > n \times 0.1$,
      - Generate bootstrapped sample D_t^* from D_t.
      Else
      - $D_t^* = D_t$.
   (ii) Randomly select $m \approx \sqrt{p}$ or $m \approx p/3$ features from D_t^*.
   (iii) Find the best split features and cutpoints using the random feature subset.
   (iv) Send down the data using the best split features and cutpoints.
   Repeat (i)-(iv) until stopping rules are met.
3. Construct trained classifiers C_b.

Test Phase :
Aggregate the B trained classifiers using simple majority voting. For a test instance x, the predicted class label from classifiers C_B is given as :

$$C_B(x) = \text{argmax}_j \sum_{b=1}^{B} I(C_b(x) = j), \text{for } \; j = 1, ..., K$$

4 Case study ('mnist' data)

The motivational experiments discussed in Sect. 2.2 show that even if the RF algorithm uses hyper-parameters that allow a maximum tree, it may under-fit the data. We extend the experiment to see that the proposed DRF algorithm relaxes the under-fitting problem. The design of the experiment is the same as described in Sect. 2.2, except that *nodesize* was set to the default value 1 to compare both methods at the maximum tree size.

The results of the experiment are summarized in Fig. 3 and Table 1. In Fig. 3, we compared the test accuracy of both methods. Boxplots in the figure are drawn using the test accuracy obtained from six repeated experiments following Larochelle et al. (2012). DRF seems significantly more accurate than RF because the boxplot of DRF does not overlap and is well above the boxplot of RF.

In Table 1, we compared the tree sizes of both methods. The values in the table are the mean and standard deviation (parenthesis) of the number of terminal nodes in 200 trees of the ensemble. The trees in DRF are on average larger than those in RF as expected. Also note that the DRF tree has higher standard deviation than the RF tree in the number of terminal nodes. This is due to the randomness that comes from bootstraps and feature subsets that occur during split rule exploration. This will eventually lead to more diverse trees.

In conclusion, for 'mnist' data, the DRF ensemble method seems to overcome the under-fitting problem of the RF ensemble method. In addition, the DRF ensemble method produces more diverse trees, which is one of the factors that make the ensemble method successful.

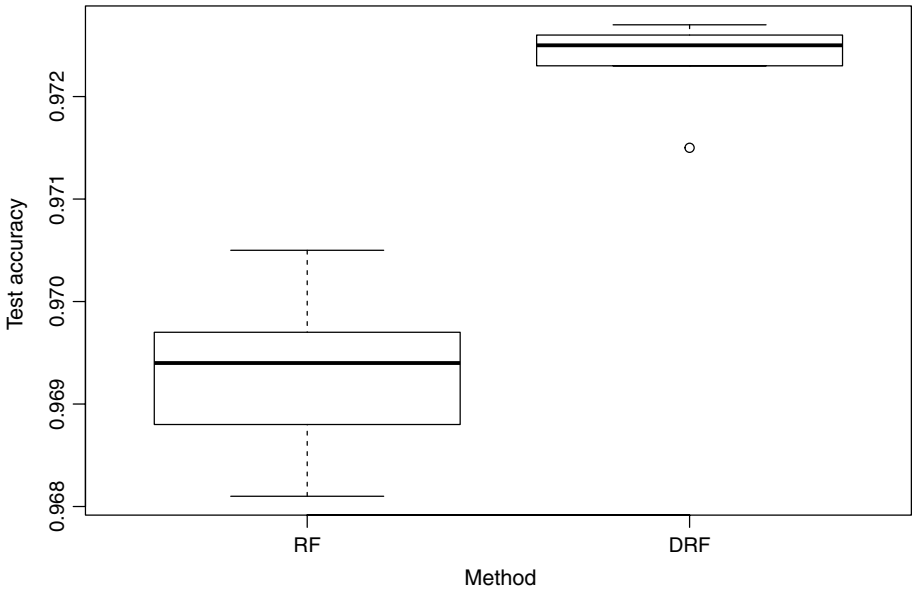

Fig. 3 Box-plot of the test accuracy of RF and DRF using 'mnist' data

Table 1 Tree size comparison between RF and DRF

Experiments	1	2	3	4	5	6
RF	4334.9	4324.1	4336.6	4327.7	4326.7	4332.2
	(87.7)	(92.2)	(101.3)	(87.8)	(91.3)	(95.5)
DRF	5979.6	5987.7	5991.7	5977.4	5955.1	5994.9
	(123.2)	(122.8)	(116.2)	(123.5)	(120.4)	(105.6)

Mean and standard deviation (parenthesis) of the number of terminal nodes of 200 trees in the ensemble

5 Experimental study

In this section, more data are used to compare the prediction accuracy of the proposed DRF method with other ensemble methods. First, we will divide 58 datasets into two groups, one for which RF may under-fit and the other for which it does not. Then, the comparison will only proceed in the data group where RF may under-fit. More details are given below.

5.1 Effect of the tree size

This section expands the discussion of the tree size effect in Sect. 2.2 with a larger number of data. The experiment was conducted on 58 real and artificial datasets that are suitable for classification problems. They mostly came from UCI data repository (Asuncion and Newman 2007) and the *mlbench* R package (Dimitriadou and Leisch 2010). The datasets are listed in Table 2.

Table 2 Descriptions of the 58 datasets

Datasets	Observations	Classes	Variables	Sources
aba	4177	2	8	UCI (Abalone)
ail	13,750	2	12	Loh (2009)
aus	690	2	14	UCI (Australian credit approval)
bal	625	3	4	UCI (Balance scale)
ban	1372	2	4	UCI (Bank note authentication)
bcw	683	2	9	Lim et al. (2000)
bld	345	2	6	UCI (BUPA liver disorders)
blo	748	2	4	UCI (Blood transfusion center)
bod	507	2	24	Kerk et al. (2003)
bos	506	3	13	UCI (Boston housing)
bre	699	2	9	UCI (Wisconsin Breast Cancer)
cir	1000	2	10	R library *mlbench* (Circle in a square)
cmc	1473	3	9	UCI (Contraceptive method choice)
col	366	3	23	UCI (Horse Colic)
cre	690	2	15	UCI (Credit approval)
cyl	540	2	30	UCI (Cylinder bands)
der	358	6	34	UCI (Dermatology)
dia	768	2	8	Loh (2009)
dna	3186	3	60	UCI (StatLog DNA)
ech	131	2	6	UCI (Echocardiogram)
eco	336	4	7	Loh (2009)
fis	159	7	6	Kim and Loh (2003)
ger	1000	2	20	UCI (German credit)
gla	214	6	9	UCI (Glass)
hea	270	2	13	UCI (StatLog heart disease)
hep	155	2	19	UCI (Hepatitis)
hil	606	2	100	UCI (Hill-valley)
imp	205	6	23	UCI (Auto imports)
int	1000	2	9	Kim et al. (2010)
ion	351	2	33	UCI (Ionosphere)
iri	150	3	4	UCI (Iris)
lak	259	6	14	Loh (2009)
led	6000	10	7	UCI (LED display)
lib	360	15	90	UCI (Libras movement)
mam	961	2	5	UCI (Mammographic mass)
mar	8777	10	4	Loh (2009)
pid	532	2	7	UCI (PIMA Indian diabetes)
pks	195	2	22	UCI (Parkinsons)
pov	97	6	6	Kim and Loh (2001)
rng	1000	2	10	R library *mlbench* (Ringnorm)
sat	6435	6	36	UCI (StatLog satellite image)
sea	3000	3	7	Terhune (1994)
seg	2310	7	18	UCI (Image segmentation)
smo	2855	3	8	UCI (Attitude towards smoking restrictions)

Table 2 (continued)

Datasets	Observations	Classes	Variables	Sources
snr	208	2	60	R library *mlbench* (Sonar)
soy	307	7	28	Loh (2009)
spa	4601	2	57	UCI (Spambase)
spe	267	2	44	UCI (SPECTF heart)
thy	7200	3	21	UCI (Thyroid disease)
trn	1000	2	10	R library *mlbench* (Threenorm)
twn	1000	2	10	R library *mlbench* (Twonorm)
usn	1302	3	27	Statlib (2010)
veh	846	4	18	UCI (StatLog vehicle silhouette)
vol	1521	6	5	Loh (2009)
vot	435	2	16	UCI (Congressional voting records)
vow	990	11	10	UCI (Vowel recognition)
wav	3600	3	21	UCI (Waveform)
zoo	101	7	16	R library *mlbench* (Zoo)

For the experiment, we used a random 50% dataset as the training set for the fitting and a random 50% dataset as the test set for the evaluation. For a reliable comparison, the entire experimental process was repeated 100 times to reduce the impact of some specific training or test sets. It can also reduce the sampling bias that can occur with an unbalanced number of class instances. All other design settings are the same as in Sect. 2.2.

The results of the experiment are summarized in Figs. 4, 5, 6, and 7 where the horizontal axis represents *nodesize* values and the vertical axis represents the relative test accuracy. Relative test accuracy is defined as (the accuracy of RF under the given *nodesize*) / (the accuracy of RF when *nodesize* is set to its default 1). Therefore, if this value is greater than 1, the RF with smaller trees is more accurate than that with the largest trees. After all, this means that RF does not under-fit. Similarly, if the value is less than 1, the RF with the largest trees is the best, which means that RF may under-fit.

Of the 58 data sets in total, we can divide the 34 data sets shown in Figs. 4 and 5 into one group and the 24 data sets shown in Figs. 6 and 7 into another. Figures 4 and 5 show that the relative test accuracies are lower than 1 for all *nodesize* values. In such datasets, the tree size of the RF is likely not large enough to provide the best performance, and there is room for further improvement. In contrast, relative test accuracy exceeds 1 in the Figs. 6 and 7. In this case, the tree size does not need to be maximum for best RF performance. In these cases, a certain level of pruning can be beneficial to RF when building the trees.

5.2 Comparison between DRF and other ensemble methods

This section compares the proposed DRF method and other classification ensemble methods, such as Bagging (Breiman 1996), Samme (a modified version of AdaBoost) (Zhu et al. 2009) and RF using the 34 datasets shown in Figs. 4 and 5. A single tree model (Therneau and Atkinson 2019) is also included in the comparison as a baseline.

The design of the experiment is as follows. As in Sect. 5.1, we used a random 50% dataset as the training set for the fitting and a random 50% dataset as the test set for evaluation. The maximum tree size without pruning was used for all ensemble methods except

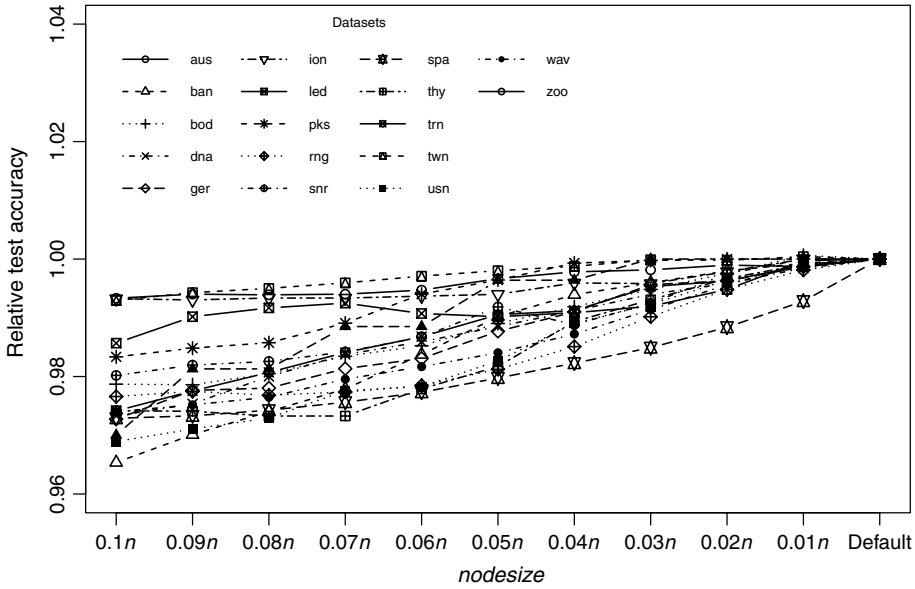

Fig. 4 Relative accuracy for each *nodesize*. RF may under-fit these dataset

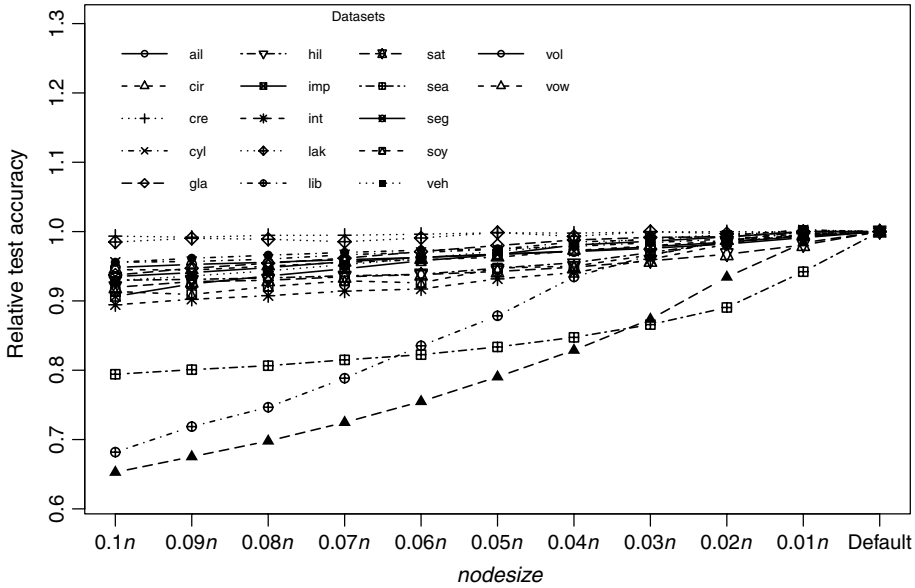

Fig. 5 Relative accuracy for each *nodesize*. RF may under-fit these dataset

Samme. For Samme, tree growth is limited by the maximum tree depth as the number of classes. as in Kim et al. (2010). The number of trees generated in the ensemble was set to 200. The single tree model uses pruning for a better accuracy. Other parameters were left

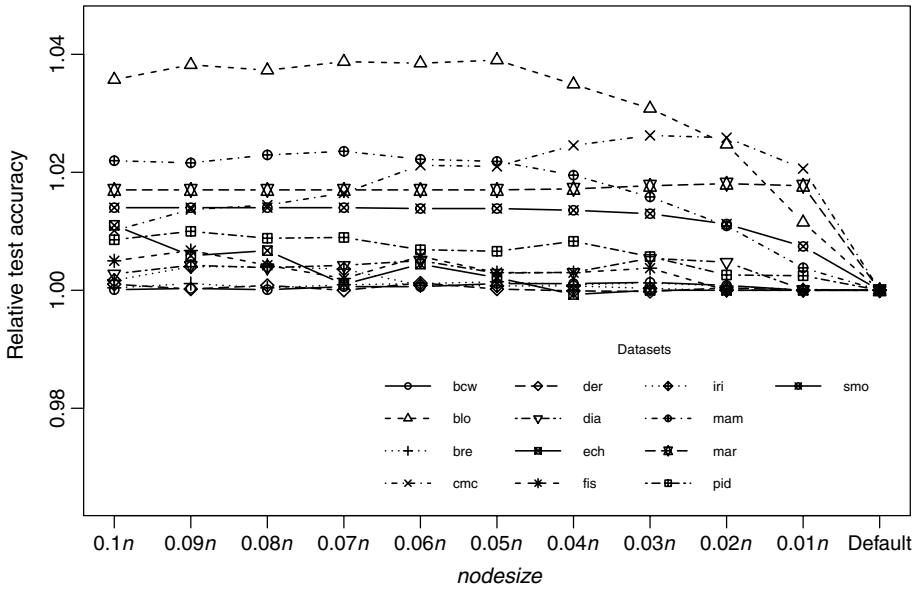

Fig. 6 Relative accuracy for each *nodesize*. RF does not under-fit these dataset

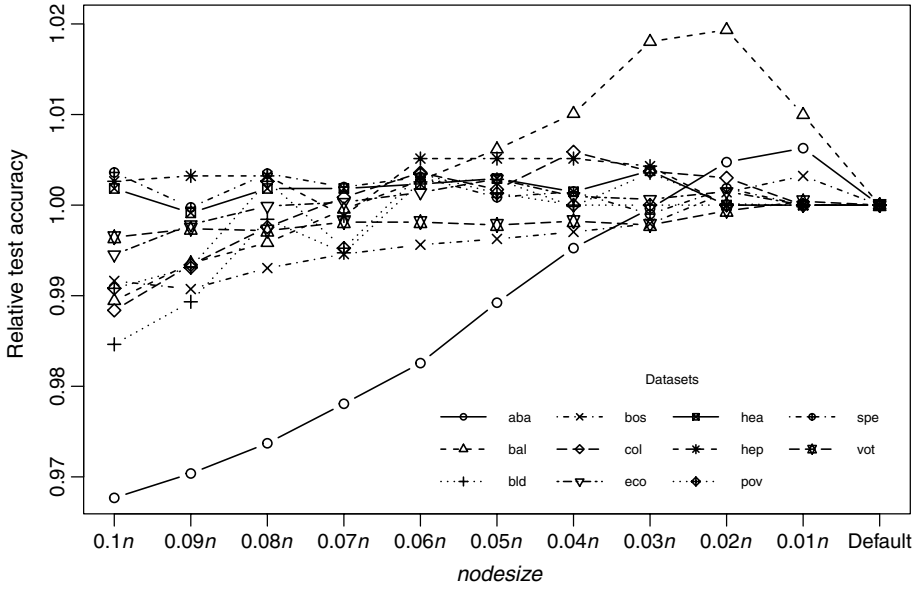

Fig. 7 Relative accuracy for each *nodesize*. RF does not under-fit these dataset

Table 3 Pairwise comparison of prediction accuracies. In each cell, results are summarized as "$a(b)$", where a is the number of dataset that the method in the column outperforms the method in the row and b is the number of dataset that the difference is statistically significant in the one-sided paired t-test

	Single tree	Bagging	Samme	RF	DRF
Single tree		32(30)	30(28)	32(31)	32(32)
Bagging	2(1)		18(16)	29(29)	32(30)
Samme	4(4)	16(9)		23(20)	26(23)
RF	2(2)	5(3)	11(8)		23(19)
DRF	2(2)	2(2)	8(7)	11(6)	

Table 4 Dominance ranks of the methods using the significant differences from the results in Table 3

	Dominance rank	Wins	Losses
DRF	87	104	17
RF	54	86	32
Samme	3	59	56
Bagging	−32	44	76
Single tree	−112	9	121

Dominance rank is defined as the number of significant wins minus the number of significant losses

to their default values. The whole experimental process was also repeated 100 times for a reliable result.

Table 3 compares the test accuracy of DRF with other methods. The values in the table are the number of times that the vertical method is more accurate than the horizontal method. For example, a value of 32(32) in the first and fifth columns means that DRF is more accurate than a single tree for 32 of 34 data sets, and the number in parentheses means the number of times it is statistically significant. On the other hand, the values in the fifth row and the first column, 2(2), indicate that there were 2 datasets for which the single tree performed better than DRF and both were statistically significant.

Table 4 indicates the ranking of the methods based on the results in Table 3. The dominance rank is defined as the number of significant wins minus the number of significant losses. For example, DRF has a dominance rank of 87 given that it had 104 significant wins and 17 significant losses. The number of significant wins for DRF is the sum of the values in the parentheses in the DRF column. Similarly, the number of significant losses for DRF is the sum of the values in the parentheses in the DRF row.

Tables 3 and 4 show that the DRF method significantly outperforms other methods when using 34 datasets whose RF tree size is not large enough. Since the RF method is the closest competitor, we conducted a more detailed comparison between DRF and RF.

Figure 8 shows the confidence interval for the difference in accuracy between DRF and RF. Of the 34 confidence intervals, 19 are above zero, which means that for 19 of the 34 datasets, DRF is significantly better than RF. Only 6 confidence intervals are located below zero.

Table 5 shows the results of the randomized complete block design (RCBD) performed to test the overall difference in accuracy between DRF and RF. The P-value for the difference between the two methods is very low, 0.0009, which means that DRF is statistically more accurate than RF.

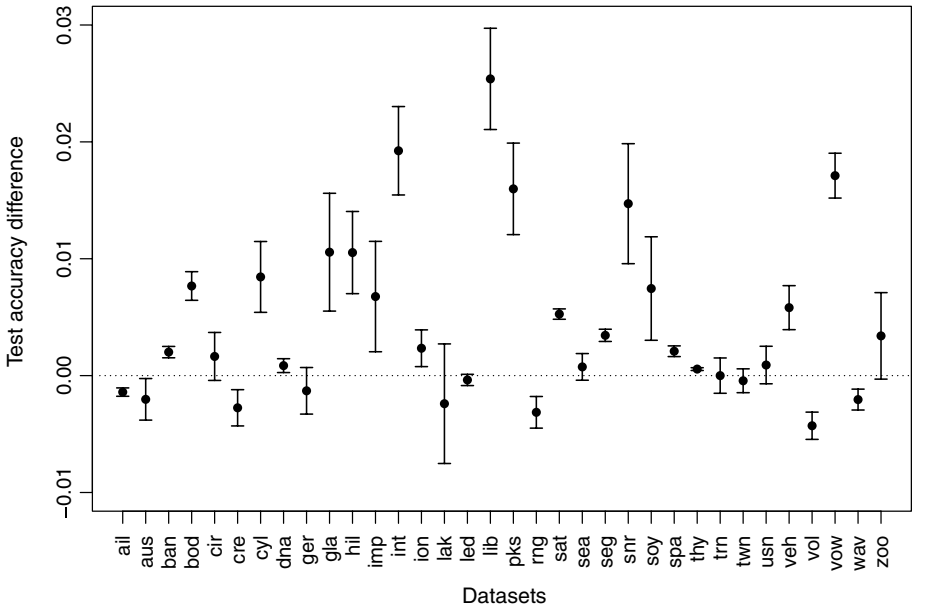

Fig. 8 Confidence intervals of accuracy differences between DRF and RF. If the confidence interval does not contain zero, the difference is statistically significant

Table 5 ANOVA table of the randomized complete block design

	Df	Sum squares	Mean squares	F value	Pr(> F)
DRF versus RF (treatment)	1	0.034	0.034	10.92	0.0009
Data (block)	33	175.43	5.316	1692.26	< 0.0001
Errors	6765	21.25	0.003		

The results so far have been compared with the prediction accuracy using test data. A comparison result was also generated using F1-score, an index that considers sensitivity and precision. F1-score is defined as $2 \times \frac{Precision \times Recall}{Precision + Recall}$. The comparison with F1-score showed little difference from previous results with test accuracy. For those interested, comparison results using F1-score are given in Appendix.

5.3 Comparison of computing times

The DRF algorithm requires more computation time, because the bootstrap technique must be used on each node of every tree in the ensemble. It also takes longer to complete, because the tree of DRF is larger than that of RF, which further increases the number of times the bootstrap must be performed.

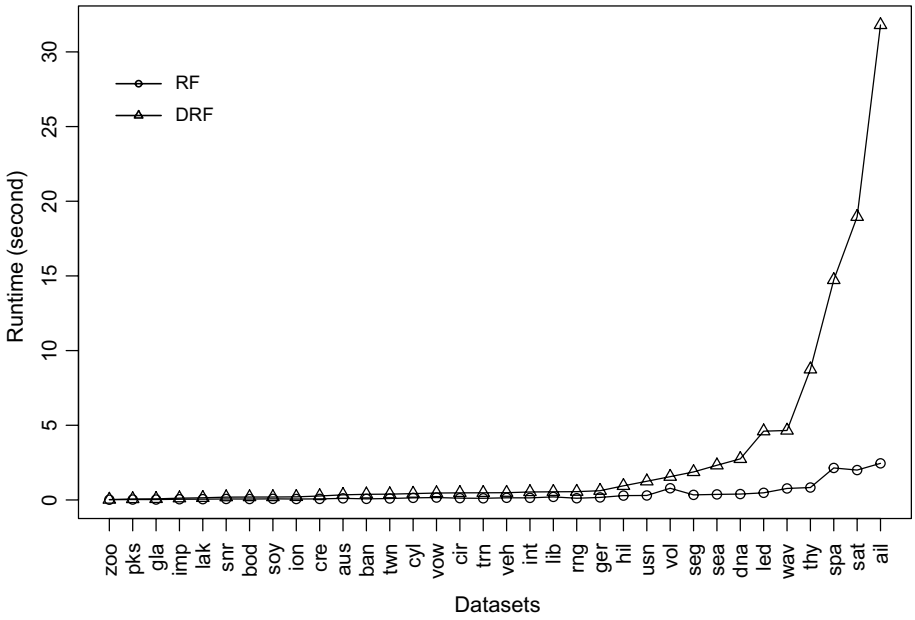

Fig. 9 Computing times of DRF and RF

Figure 9 shows the runtime of the two methods for each of the 34 data sets. The runtime did not make a big difference in small data sets, but the gap is widening in large data sets as expected.

5.4 Comparison of DRF and tuned RF (instance size)

Is there any other way to make the trees bigger in RF method? If so, it may exhibit similar accuracy to DRF. One possible way is to use instances which is sampled without replacement, instead of using bootstrap (sampling with replacement) in each tree. Bootstrapping gets n instances, but since it uses sampling with replacement, the number of unique instances is about $0.632 \times n$. If $0.9 \times n$ instances are sampled without replacement, the trees of RF will be larger because there are more unique instances. However, as the number of unique instances increases, the pool of data used to construct each tree becomes more similar. This may lead to weaken the diversity of the RF trees. As a related study, Martínez-Muñoz and Suárez (2010) reported the effect of instance size on Bagging and showed that sampling without replacement, rather than bootstrap, is more effective on some data set.

In the experiment of this section, when fitting the RF method, bootstrapping was disabled and the *samplesize* parameter was set to $0.9 \times n$ to make the RF trees bigger. As a result, it was found that trees are about 1.26 times larger than those of the typical RF. However, this method was more accurate on only 18 of the 34 datasets. In the rest of the data set, the typical RF was more accurate. In short, the accuracy of RF has not improved just for the reason that the generated trees are larger. This is because the diversity of classifiers is weakened by creating trees with more similar instances throughout the ensemble (Martínez-Muñoz and Suárez 2010).

Table 6 Pairwise comparison of tuned RF (instance size) and DRF

	RF	Tuned RF (instance size)	DRF
RF		18(15)	23(19)
Tuned RF (instance size)	0(0)		20(12)
DRF	11(6)	14(7)	

In each cell, results are summarized as "$a(b)$", where a is the number of dataset that the method in the column outperforms the method in the row and b is the number of dataset that the difference is statistically significant in the one-sided paired t-test. Data with the same accuracy were excluded from the counting

It is of interest to compare the accuracy of DRF and the tuned RF (instance size), where the tuned RF is using *sampleize* which gives more accurate results for each data. As shown in Table 6, if the RF is tuned to the instance size, the difference in accuracy from DRF is reduced. However, DRF is still superior to tuned RF (instance size). The reason may be that in the case of DRF, the diversity of classifier is not weakened because bootstrapping is performed at every node of each tree.

5.5 Comparison of tuned RF and tuned DRF

It is known that the number of candidate features in each node of a tree affects the accuracy of the RF ensemble. The number of candidate features, known as *mtry* in the randomForest package of R, is a hyper-parameter that users can choose. In general, the most commonly used value for *mtry* is $\sqrt{p}$, where p is the number of all features (Han and Kim 2019). The results shown in Sects. 5.2 and 5.4 were completed using $\sqrt{p}$ as the default value. However, the default does not always guarantee the highest accuracy, and depending on the data, more favorable values may exist.

In this section, we compare the performance of tuned RF and tuned DRF under the situation where their performances are maximized by appropriately selecting the *mtry* values for each data. The grid search method was used to select *mtry* values. In addition, for RF, tuning for instance size was also performed as in Sect. 5.4.

The pairwise comparison of prediction accuracy is given in Table 7. When comparing RF and tuned RF, data with the same accuracy were excluded from counting. This is the same for DRF. According to Table 7, the tuned DRF exceeds the performances of RF, tuned RF, and DRF.

Table 7 Pairwise comparison of tuned RF and tuned DRF

	RF	Tuned RF	DRF	Tuned DRF
RF		28(22)	23(19)	27(22)
Tuned RF	0(0)		16(11)	20(12)
DRF	11(6)	18(12)		15(11)
Tuned DRF	7(3)	14(7)	0(0)	

In each cell, results are summarized as "$a(b)$", where a is the number of dataset that the method in the column outperforms the method in the row and b is the number of dataset that the difference is statistically significant in the one-sided paired t-test. Data with the same accuracy were excluded from the counting

6 Conclusion

Random forest (RF) is known to better predict when using larger trees without pruning. Therefore, it is common to create trees of the maximum size possible. In this paper, we found that the maximum sized tree produced by RF may not be large enough for some dataset. This indicates that RF may underfit, even with the most favorable option. To cope with this, we developed a method called double random forest (DRF) that creates trees larger than RF.

The DRF ensemble method has two distinctions from RF. First, DRF uses the whole training data to grow a decision tree. Since whole training data has more unique instances than the bootstrap data, it tends to make the tree bigger than the tree in RF. For data where RF under-fits, this approach can remedy the problem, reducing the prediction bias. Second, DRF uses the bootstrap sampling only to find the partitioning rule for each intermediate node. That is, it searches for the best partitioning rules by using a random bootstrap of instances and a random subset of features on each node of the tree. Once the partitioning rule is found, the bootstrap data is restored to the original data and the instances are sent down to the child node. We can argue that DRF creates more diverse tree ensemble by adding randomness to the partitioning rules with a bootstrap and a feature subset.

Experimental studies using 34 real or artificial data sets where RF under-fit show that the proposed DRF method is far superior to other classification ensemble. Moreover, we found that DRF is statistically better than RF in terms of test accuracy through the analysis of randomized complete block design (RCBD). This trend was the same for both RF and DRF tuned for mtry values. However, it is true that the DRF algorithm is more complex and takes longer time to fit the model compared to RF.

For those datasets where the ideal nodesize is not 1, the accuracy of the DRF and RF methods was similar and no statistical difference was found. That is, when RF does not under-fit, we do not think that DRF is needed because trees do not need to be longer.

Acknowledgements Hyunjoong Kim's work was supported by Basic Science Research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (No. 2016R1D1A1B02011696). Yung-Seop Lee's work was supported by National Research Foundation of Korea, Grant Number 2017R1E1A1A03070102.

Appendix

The pairwise comparison result with F1-score is given in Table 8. It can be seen that using the F1-score leads to the same conclusions as in Sect. 5.2.

Table 8 Pairwise comparison of F1-scores

	Single tree	Bagging	Samme	RF	DRF
Single tree		31(30)	29(28)	30(30)	31(31)
Bagging	3(2)		17(15)	27(26)	30(26)
Samme	5(5)	17(14)		24(17)	24(20)
RF	4(3)	7(3)	10(9)		23(18)
DRF	3(3)	4(2)	10(7)	11(6)	

In each cell, results are summarized as "$a(b)$", where a is the number of dataset that the method in the column outperforms the method in the row and b is the number of dataset that the difference is statistically significant in the one-sided paired t-test

Software: The R code that implements the method and experiments presented in this article is provided in 'https://github.com/shan-stat/DRF'.

References

Amaratunga, D., Cabrera, J., & Kovtun, V. (2008). Microarray learning with ABC. *Biostatistics*, *9*, 128–136.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository. University of California, Irvine, School of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Banfield, R., Bowyer, K., Kegelmeyer, W., & Hall, L. (2007). A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 173–180.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*, 105–139.

Boulesteix, A. L., Janitza, S., Kruppa, J., & König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 496.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794.

Dietterich, T. G. (2000). *Ensemble methods in machine learning*. Berlin: Springer.

Dimitriadou, E., & Leisch, F. (2010). *mlbench: Machine learning benchmark problems*. Vienna: R Foundation for Statistical Computing.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the thirteenth international conference on machine learning*, 148–156.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computational System Sciences*, *55*, 119–139.

Freeman, E. A., Moisen, G. G., Coulston, J. W., & Wilson, B. T. (2015). Random forests and stochastic gradient boosting for predicting tree canopy cover: comparing tuning processes and model performance. *Canadian Journal of Forest Research*, *46*(3), 323–339.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. New York: Springer.

Han, S., & Kim, H. (2019). On the optimal size of candidate feature set in Random forest. *Applied Sciences*, *9*, 898.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 993–1001.

Hastie, T., James, G., Tibshirani, R., & Witten, D. (2013). *An introduction to statistical learning : With application in R*. New York: Springer.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *10. The elements of statistical learning (2nd ed.)*. New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0.

Hernandez-Lobato, D., Martinez-Munoz, G., & Suarez, A. (2013). How large should ensembles of classifiers be? *Pattern Recognition*, *46*, 1323–1336.

Huang, B. F. F., & Paul, C. B. (2016). The parameter sensitivity of random forests. *BMC Bioinformatics*, *17*, 331.

Kerk, C. J., Heinz, G., Johnson, R. W., & Peterson, L. J. (2003). Exploring relationships in body dimensions. *Journal of Statistic Education*, 11. http://www.amstat.org/~publications/jse/v11n2/datasets.heinz.html.

Kim, H., & Loh, W. Y. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistics Association*, *96*, 589–604.

Kim, H., & Loh, W. Y. (2003). Classification trees with bivariate linear discriminant node models. *Journal of Computational and Graphical Statistics*, *12*, 512–530.

Kim, H., Kim, H., Moon, H., & Ahn, H. (2010). A weight-adjusted voting algorithm for ensemble of classifiers. *Journal of Korean Statistics Society*, *40*, 437–449.

Larochelle, H., Mandel, M., Pascanu, R., & Bengio, Y. (2012). Learning algorithms for the classification restricted Boltzmann machine. *Journal of Machine Learning Research*, *13*(1), 643–669.

Lim, T. S., Loh, W. Y., & Shih, Y. S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, *40*, 203–228.

Lin, Y., & Jeon, Y. (2012). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, pp. 578–590.

Loh, W. Y. (2009). Improving the precision of classification trees. *The Annals of Applied Statistics*, *3*, 1710–1737.

Martínez-Muñoz, G., & Suárez, A. (2010). Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, *43*, 143–152.

Mason, L., Baxter, J., Bartlett, P. L., & Frean, M. (1999). Boosting algorithms as gradient descent. In S. A. Solla, T. K. Leen, K. Müller (Eds.). *Advances in neural information processing systems 12*. MIT Press, Cambridge, pp. 512–518.

Oshiro, T., Perez, P., & Baranauskas, J. (2012). How many trees in a random forest? In *International workshop on machine learning and data mining in pattern recognition* (pp. 154–168). Berlin: Springer.

Probst, P., & Boulesteix, A.-L. (2018). To tune or not to tune the number of trees in a random forest? *Journal of Machine Learning Research*, *18*, 1–18.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Statlib. (2010). Datasets archive. Carnegie Mellon University, Department of Statistics. http://lib.stat.cmu.edu.

Terhune, J. M. (1994). Geographical variation of harp seal underwater vocalizations. *Canadian Journal of Zoology*, *72*, 892–897.

Therneau, T., & Atkinson, B. (2019). Recursive partitioning and regression trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart.

Wolf, B. J., Hill, E. G., & Slate, E. H. (2010). Logic forest: An ensemble classifier for discovering logical combinations of binary markers. *Bioinformatics*, *26*, 2183–2189.

Zhu, J., Zou, H., Rosset, S., & Hastie, T. (2009). Multi-class AdaBoost. *Statistics and its Interface*, *2*, 349–360.